UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE AND ENGINEERING

FINAL EXAMINATION, DECEMBER 2000

CSC444F - SOFTWARE ENGINEERING I
Examiner: Steve Easterbrook

Exam type: A (Closed Book; no aids permitted)
Calculator type: 1 (any programmable and non-programmable calculator)

Duration: 2.5 hours.

This exam paper consists of 8 questions on 4 pages.
Answer ANY 5 questions. All questions have equal weight (20 marks per question)

QUESTION 1 (Testing)

a) Define *black box* and *white box* testing. What are the advantages of each approach? Why are both necessary? [5 marks]

b) Why is it important to be able to partition the test space into *equivalence classes*? For the following code fragment, describe 3 different test cases, and for each, describe the class of test cases it represents. [5 marks]

```
char * triangle (int x, y, z) {
/* requires: The parameters are in ascending order (i.e. x <= y <= z)
   effects: If x, y and z are the lengths of the sides of a triangle,
   this function classifies the triangle using one of the three
   strings, "scalene", "isosceles" or "equilateral". If x, y, and z
   do not form a triangle, the empty string is returned.
*/
   char *r;
   r="scalene";
   if (x==y || y==z)
      r="equilateral";
   if (x==z)
      r="isosceles";
   if (x <= 0 || (x+y) <= z)
      r="";
   return (r); }
```

c) *Full path coverage* testing requires that every possible *path* through the code be tested at least once. Why is full path coverage testing desirable? For the code fragment above, how many test cases would be needed for full path coverage? Why might full path coverage be impossible to achieve for some programs? [3 marks]

d) *Decision point coverage* testing requires that each outcome of each decision point be tested at least once. Why does this criteria usually require fewer test cases than full path coverage? What kinds of error might this testing miss? [2 marks]

e) Boeing estimates that half the cost of the software development for the 777 aircraft was spent on testing, including regression testing. How might this cost be reduced without sacrificing quality? [5 marks]

[TURN OVER]

QUESTION 2 (Software Architectures)

a) A software architecture describes a high-level design view of a software system. What are the advantages of explicitly describing the architecture independently from the implementation? [2 marks]

b) Pipe-and-filer architectures treat a task as a series of processing steps, such that the output of one step forms the input of the next step. Give an example of a pipe-and-filter system. What are the advantages and disadvantages of this style? [4 marks]

c) Object oriented architectures can be difficult to develop because each object needs to know which other objects exist and how to call their methods. This limitation can be overcome using implicit invocation. Describe how an implicit invocation scheme works, and give two examples of its use. [4 marks]

d) An *architectural description language (ADL)* is a general purpose language for describing software architectures. What kinds of basic components and basic connectors would you expect an ADL to provide? What advantages are there in using an ADL? [4 marks]

e) The Free Software Foundation proposes to develop an open source suite of tools for developing and debugging Java programs. What architectural styles would you consider for this project, and what aspects of this project would affect your choice of style? [6 marks]

QUESTION 3 (Software Inspections)

a) Software inspections are widely used for verification. What are the benefits of inspection? What kinds of errors will inspection catch that testing cannot? [4 marks]

b) Some companies exclude all management from code inspections. What are the advantages and disadvantages of this rule? [2 marks]

c) Briefly describe each of the following approaches for structuring an inspection meeting. What are the advantages of each?
   i) Round robin;
   ii) Using a checklist;
   iii) Conducting a walkthrough. [6 marks]

d) What factors reduce the effectiveness of inspections? What can be done to address these problems? [4 marks]

e) Imagine you work for an aerospace company that used to carry out Fagan inspections of program code, but which now uses an automatic code generation tool that generates programs from design notations. The output of the tool is hard to understand as it is not commented, and is poorly structured. For this reason the company has cancelled all code inspections and plans to rely on testing as its main verification technique. What are the advantages and disadvantages of this decision? [4 marks]

[TURN OVER]

QUESTION 4 (Requirements Engineering & Specification)

a) Zave defines Requirements Engineering as *"the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems [and] the relationship of these factors to precise specifications of software behaviour, and to their evolution over time and across software families"*. Why is Requirements Engineering considered to be the most important part of software engineering? [4 marks]

b) Requirements should state what a system should do, without stating how it should do it. Why is this distinction useful? [2 marks]

c) *Structured Analysis* proceeds by modelling the current physical system, abstracting a model of the current logical system, and then modelling the new logical system. What are the advantages and disadvantages of building these three separate models? What representations are used for each of these models? [4 marks]

d) Explain why each of the following is an important property of a software specification, and explain how it can be achieved when writing specifications:
   i) validity;
   ii) traceability;
   iii) verifiability. [6 marks]

e) Project managers sometimes regard work put into writing high quality specifications as "gold plating", and claim that it is unnecessary as it doesn't contribute to producing program code. Under what circumstances is this view sensible, and under what circumstances is it foolish? In the latter case, how would you persuade such a manager that the specification does need to be high quality? [4 marks]

QUESTION 5 (Software Maintenance)

a) Why is software maintenance difficult? Why is it necessary? [3 marks]

b) Why does a "quick-fix" approach to maintenance rapidly degrade the quality of a software system? What alternatives are there? [4 marks]

c) Software maintenance activities can be classified as *corrective* (fixing errors), *adaptive* (responding to change) and *perfective* (improving the original software). Why is this distinction useful? How would you expect the proportion of time spent on each activity to change as the software ages? [4 marks]

d) Lehman's laws of software evolution identify a number of regularities, including:
   i) *Conservation of organisational stability*, which states that the work output (e.g. new functions added per month) of a development team is roughly constant over the life of a software system.
   ii) *Conservation of familiarity* which states that the amount of new functionality in each release of a system is roughly constant.

   What are the causes of these regularities? Under what circumstances might these laws not hold? [4 marks]

e) Your company has just acquired a smaller company that sells office automation software. The smaller company's spreadsheet software has a large market share, with many satisfied users (A major reason for the acquisition was that these existing users are potential customers for your company's other products). Unfortunately, no documentation for the spreadsheet software can be found, and the source code is not commented. How would you go about maintaining this software to keep the customers happy? [5 marks]

[TURN OVER]

QUESTION 6 (Data Abstraction)

a) A *data abstraction* describes a data structure and the set of operations that can be performed upon it. What are the advantages of data abstractions? [3 marks]

b) A data abstraction is said to be *adequate* if it provides all the operations on the encapsulated data that other programmers will need. Why is adequacy hard to test? Why does adequacy not guarantee that the abstraction will be convenient to use? [4 marks]

c) Once a data abstraction has been defined, it should be possible to change the implementation of the abstraction without affecting the *correctness* of the rest of the program. What *will* it affect? When should you consider changing the implementation of a data abstraction? [4 marks]

d) What is the difference between *object-oriented design* and *object-oriented programming*? Explain how object oriented designs can be implemented in a procedural programming language. [4 marks]

e) Object oriented programs are claimed to be more maintainable than structured programs. How would you go about testing this claim experimentally? What factors do you think might affect the validity of your experiment? [5 marks]

QUESTION 7 (Project Management)

a) DeMarco states that "you cannot control what you cannot measure". What does this mean for software project managers? [2 marks]

b) A project manager can modify three basic elements of a software project: the resources available, the time available, and the amount of product to be built. Describe how each of these three can be varied during a development process in order to ensure the resulting software is of high quality. [6 marks]

c) Risk management is an essential part of project management. Describe three typical risks that can occur in a software project, and for each suggest two possible countermeasures. [6 marks]

d) Brooks argues that adding people to a project that is running late will make it even later. Why would this be the case? [2 marks]

e) Your manager asks you to develop a software metrics collection program for the company. What questions should you ask before you suggest any specific metrics to collect? [4 marks]

QUESTION 8 (Software Process Modelling)

a) Leon Osterweil published a famous paper entitled "*Software Processes Are Software Too*", in which he argued that software development processes could be modelled at a fine level of detail, as algorithms or even programs. What are the advantages of modelling software development processes to this level of detail? What are the difficulties? [4 marks]

b) The *Capability Maturity Model* (CMM) rates software companies according to how well they identify and manage their software processes. The model has five levels: Initial, Repeatable, Defined, Managed, and Optimising. Briefly describe each of the five levels. What advantages are there for a company to move up to the top level? [6 marks]

c) Why is process improvement unlikely to occur unless an organisation defines and manages its processes? [2 marks]

d) What notations would you use to define your software development process? What are the advantages and disadvantages of these notations? [4 marks]

e) The company you work for is considering instituting a company-wide software process modelling effort, and plans to set up a Software Engineering Process Group to take responsibility for this effort. What advice would you give to this group to help ensure its success? [4 marks]